Lab Assignment

Cybersecurity Professional Program

Introduction to Python for Security

Functions

PY-05-L1 Calculator

Lab Objective

Practice creating functions, defining parameters, and performing mathematical calculations in Python.

Lab Mission

Create a function to handle calculation operations.

Lab Duration

15–25 minutes

Requirements

- Basic knowledge of Python

Resources

- Environment & Tools
  - Windows/Linux
    - PyCharm
    - Python 3

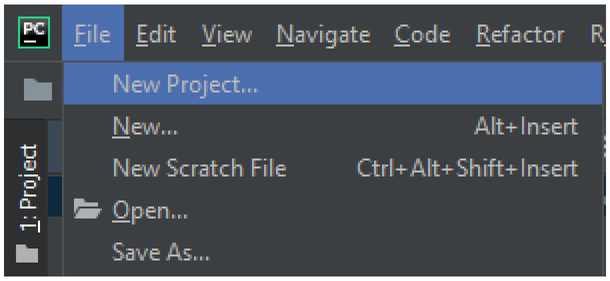
Textbook References

- Chapter 5: Functions
  - Section 1: Introduction to Functions

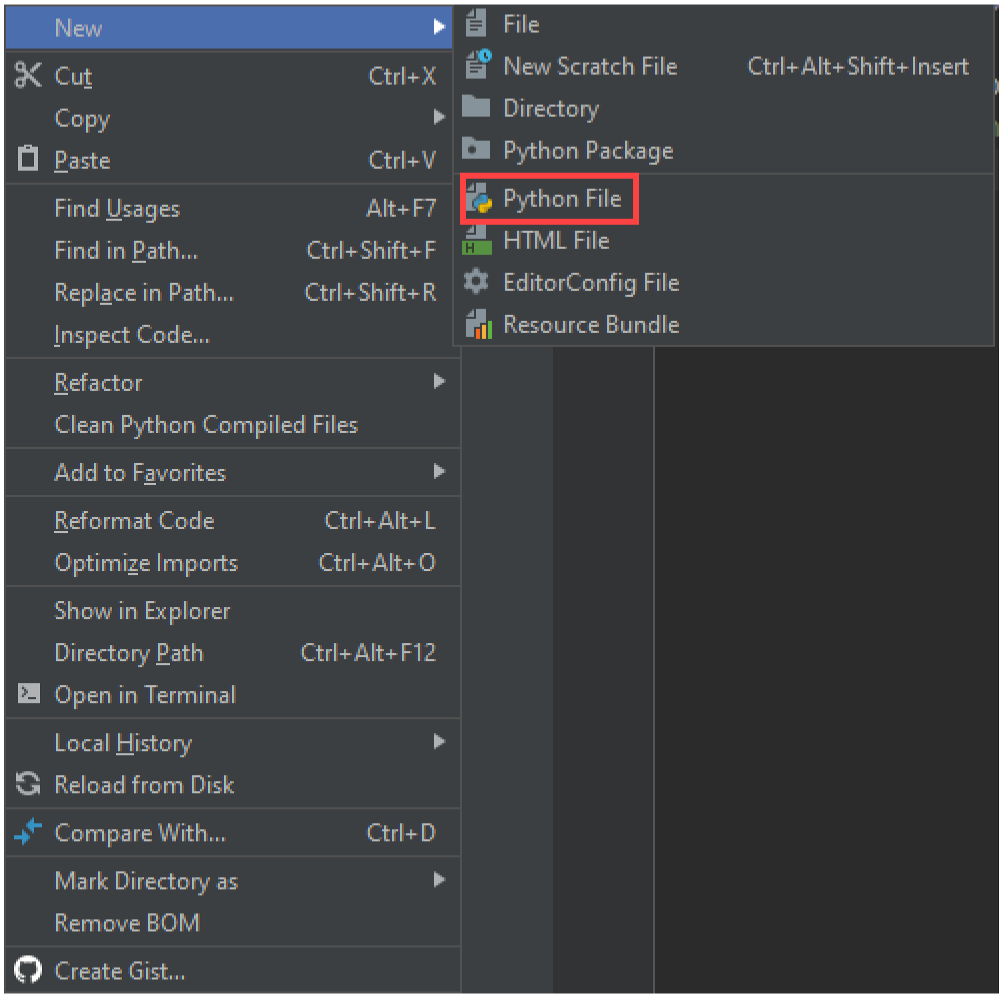
Lab Task: Creating a Calculator with Python

In this task, you will create a program that performs simple mathematical calculations and is separated into functions.

1 Open PyCharm, click **File** at the top left, and select ***New Project...***.



2 Create a new Python file in PyCharm by right-clicking the project you created and selecting **New** > ***Python File***.



3 Request from the user two numbers and an operator.

```
first_num = int(input("Please enter the first number: "))
second_num = int(input("Please enter the second number: "))
operator = input("Please enter one of the following operators: +,
-, *, / :")
```

4 Define a function that accepts two parameters for addition.

```
def add(num1, num2):
```

5 Configure the function to return the result of the addition and its description.

```
def add(num1, num2):
    description = "{} + {} ".format(num1, num2)
    return "The result of {} = {}".format(description, num1 +
num2)
```

6 Create a similar function to perform a subtraction operation.

```
def sub(num1, num2):
    description = "{} - {} ".format(num1, num2)
    return "The result of {} = {}".format(description, num1 - num2)
```

7 Create a similar function to perform a multiplication operation.

```
def mult(num1, num2):
    description = "{} * {} ".format(num1, num2)
    return "The result of {} = {}".format(description, num1 * num2)
```

8 Create a similar operation to perform a division operation.

```
def div(num1, num2):
    description = "{} / {} ".format(num1, num2)
    return "The result of {} = {}".format(description, num1 / num2)
```

9 Create the main function that will handle the execution of the calculation commands and print the result.

```
def calc():
```

10 Create a dictionary to connect the selected parameter and the appropriate function.

```
def calc():
    allowed_calculations = {"+": add, "-": sub, "*": mult, "/": div}
```

11 In the main function, allow the execution of one of the calculation functions to be performed according to the selected parameter and print the result.

```
def calc():
    allowed_calculations = {"+": add, "-": sub, "*": mult, "/": div}

    result = allowed_calculations[operator](first_num, second_num)
    print(result)
```

12 Conclude the function's execution from the dictionary with a ***try*** block. This is used to capture errors of division by 0 and unallowed parameters.

```
def calc():
    allowed_calculations = {"+": add, "-": sub, "*": mult, "/": div}

    try:
        result = allowed_calculations[operator](first_num, second_num)
        print(result)
```

13 Add an exception to capture unallowed parameters and print an appropriate message.

```
def calc():
    allowed_calculations = {"+": add, "-": sub, "*": mult, "/": div}
    try:
        result = allowed_calculations[operator](first_num, second_num)
        print(result)

    except KeyError:
        print("The parameter doesn't exist.")
```

14 Add an exception to capture division by zero and print an appropriate message.

```
def calc():
    allowed_calculations = {"+": add, "-": sub, "*": mult, "/": div}
    try:
            result = allowed_calculations[operator](first_num, second_num)
            print(result)

    except KeyError:
        print("The parameter doesn't exist.")
    except ZeroDivisionError:
        print("Can't divide by 0.")
```

15 Invoke the main method to run the program.

```
calc()
```